**Apostila :** aplicando orientação a objetos em Aplicações Web com turmas de ensino técnico profissionalizante.

*Competência:* ambientar IDE ou Editor para desenvolvimento de aplicativos web

# Introdução

Os paradigmas de orientação a objetos é um dos paradigmas mais utilizado no mundo da tecnologia da informação. Ele traz conceitos que se aproximam do mundo real. O paradigma tem uma “forma de pensar” e isso demanda tempo e dedicação de estudo. Em uma aula de orientação a objetos o aluno não irá todo o paradigma. É preciso tempo, e cada uma vai criar sua própria “nárnia” e viajar no paradigma entendendo-o em um tempo oportuno. Paciência é a palavra-chave, que auxilia tanto o professor como o aluno de orientação a objetos. Nesses vinte anos de educador em cursos técnicos e profissionalizantes e aperfeiçoamento com grandes empresas, aprendi que essa fase geralmente é ignorada e a parte teoria as vezes distante da prática. Esse artigo irá abordar uma técnica que criei nesses vinte anos de experiência. A técnica consiste em uma metodologia de educação por competência.

# Tecnologias

Escolhi a linguagem python, como a tecnologia principal e com ela diversos frameworks e outra tecnologias web de front-end como Java Script, HTML, CSS e Bootstrap. O framework web será um micro-framework chamado Flask. Nele teremos duas áreas , a primeira será estática, onde iremos explorar as tecnologias tradicionais de aplicações web e a segunda , a dinâmica, também chamada de backend, onde o flask, sqlalchemy fará a integração com um banco de dados realizando o famoso CRUD. Essas tecnologias foram escolhidas pela facilidade para iniciantes e por fazer parte de uma “stack” forte para o mercado de trabalho. Iremos implementar algumas bibliotecas python para ciência de dados e inteligência artificial colocando assim uma “cereja” no bolo desse artigo, levando o aluno a entender a aplicação do paradigma em uma visão 360º.

# Hello World POO

A sigla POO, significa programação orientação a objetos, e é um paradigma , uma forma de pensar e de ver o mundo ao redor, transformando-o em abstração tecnológica, representado por classes que possuem um conjunto de objetos, define-se atributos que são as características do objeto e funcionalidades, que são os métodos, ou seja, as ações desses objetos. Lendo esse texto, você pode entender a POO como um “organismo vivo” que necessita de padrões, por isso, define-se também, herança entre as classes e interfaces para padronizar os comportamentos desses objetos. Nesse mundo tão incrível vamos as etapas de ambientação de nossas tecnologias web priorizando nosso primeiro Hello World POO.

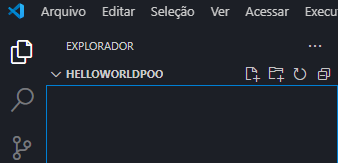
Esse artigo considera que você já conhece a linguagem python e configurou as ferramentas python, flask e tem um conhecimento básico de HTML, CSS, Java Script.

# Ambientação

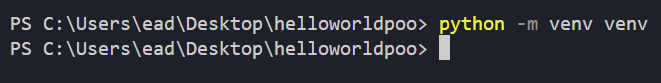
Iniciemos nosso “Hello World POO” abrindo um projeto básico flask. Você deverá rodar a aplicação e terá uma página index.html que será renderizada por uma rota “/”. Ao executar a aplicação deverá aparecer o seguinte resultado: **“Hello World POO”.** A nossa primeira competência é: “ambientar uma aplicação web flask”. Essa competência tem algumas habilidades. Entenda competência como um objetivo geral de estudo e habilidades como partes pequenas desse objetivo geral. A competência respeita seu tempo de aprendizagem e utiliza-se de conhecimentos e atitudes também chamadas de softskills.

A primeira habilidade é instalar as ferramentas do nosso projeto web

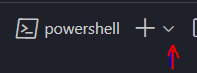
Inicie criando uma pasta e abrindo no Visual Studio Code:



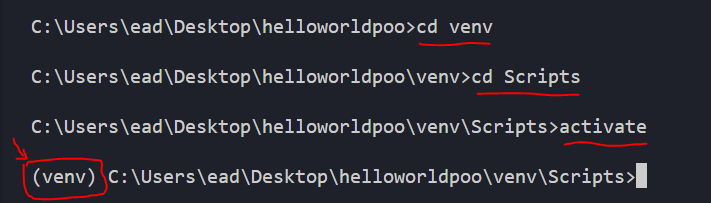
Abra o terminal e crie um ambiente virtual python:



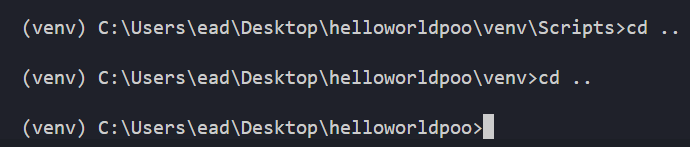
No terminal clique na setinha e habilidade “**command prompt**”



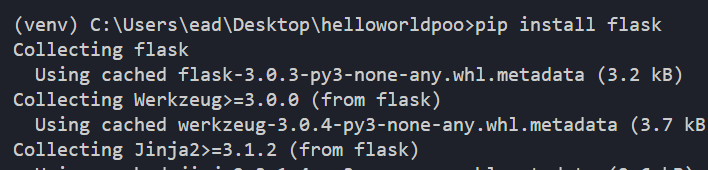
Agora você vai ativar a “venv” o ambiente virtual python



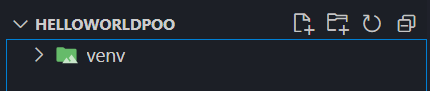
Depois você deve sair da pasta



Próximo passo é instalar o micro framework web Flask



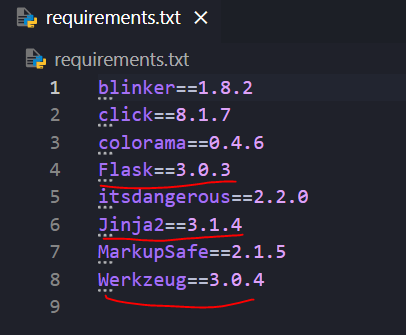
Verifique agora, se apareceu uma pasta, **chamada venv**



Não esqueça de criar o requirements.txt



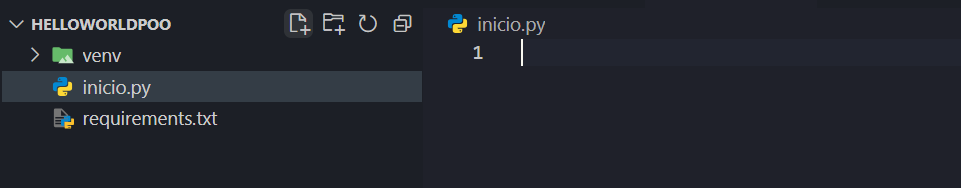
Abra o arquivo e visualize os pacotes instalados



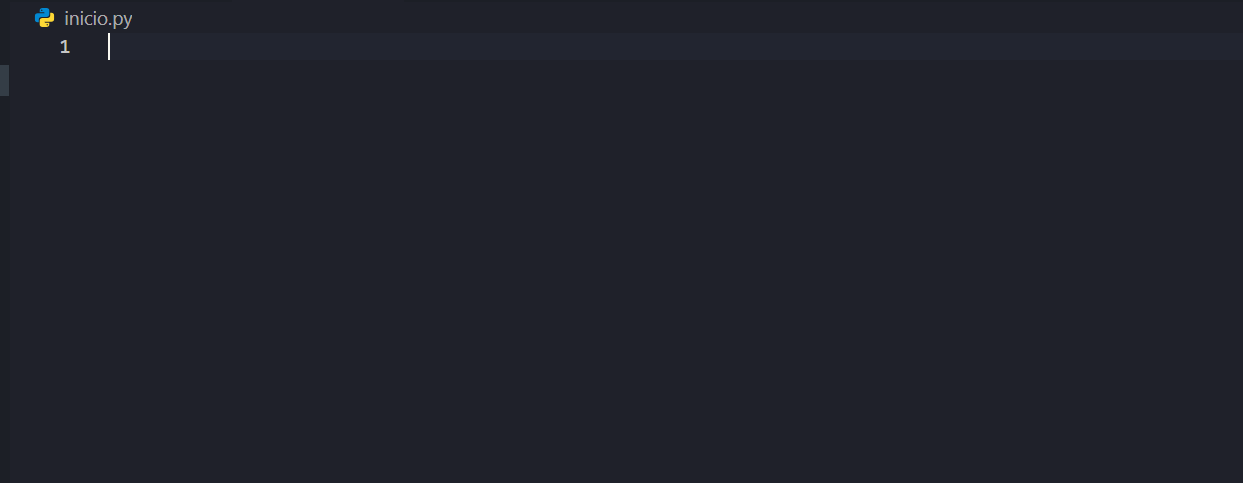
Esses que estão marcados merecem nossa atenção:

* **Flask 3.0.3** : Micro Framework Web
* **Jinja 2 3.1.4** : Template Engine
* **Werkzeug 3.0.4** : Responsável pelo funcionamento do Client/Server HTTP

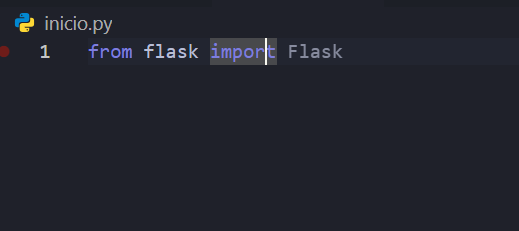
Crie um arquivo inicio.py, para iniciarmos nossa codificação:



Abra e se posicione na área do editor de código

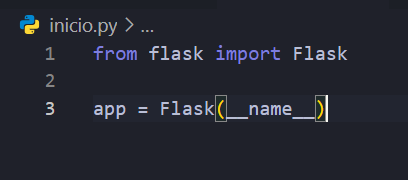


Toda classe inicia com a letra maiúscula seguindo um padrão PascalCase



O primeiro “flask” minúsculo é um módulo, já o seguindo “Flask” é uma classe. Esse padrão auxilia o programador em uma leitura rápida reconhecer a existência de classes dentro de um projeto.

Agora vamos criar um objeto da classe Flask



App é um objeto da classe Flask. Quando definimos Flask( ) , com os parênteses estamos chamando o construtor. O construtor é um método especial que cria e inicializa o objeto na memória. Ele é muito importante na orientação a objetos e vamos mais a frente retornar nesse conceito de construtor.

Na sequência, vamos criar uma rota. Imagine que você cria um site [www.turmadocodigo.com.br](http://www.turmadocodigo.com.br), e você deseja mostrar os alunos dessa turma, então, você irá criar uma rota:

[www.turmadocodigo.com.br/alunos](http://www.turmadocodigo.com.br/alunos)

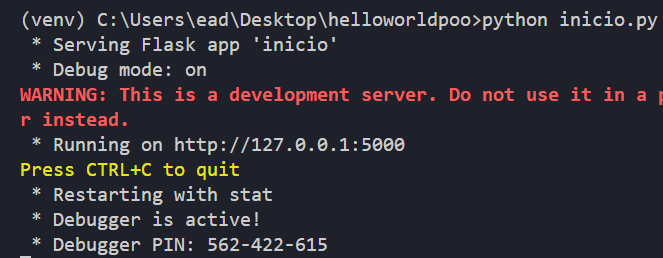
A rota /alunos irá mostrar uma lista de alunos cadastrados no sistema.

**Definir rotas** em uma aplicação web envolve mapear URLs específicas para funções ou controladores que processarão essas requisições.

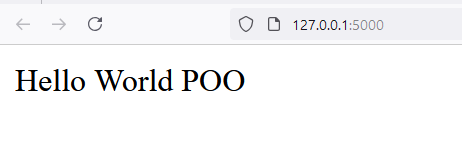
Vamos criar a nossa rota “/” para executar uma mensagem na primeira página da aplicação web



Execute uma aplicação Flask , executando via terminal o arquivo inicio.py



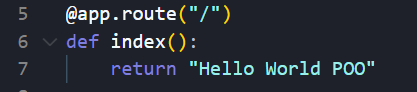
Clique em <http://127.0.0.1:5000> e execute a aplicação web.



Temos nossa aplicação web.

Agora vamos voltar ao código e ver alguns conceitos de orientação a objetos:

**Analisando o Código**



**Linha 5**: definição de uma anotação que “anota” uma função definindo-a como uma rota.

**Linha 6**: definição de uma função python

**Linha 7**: retornando um texto “Hello World POO” na função **index( )**



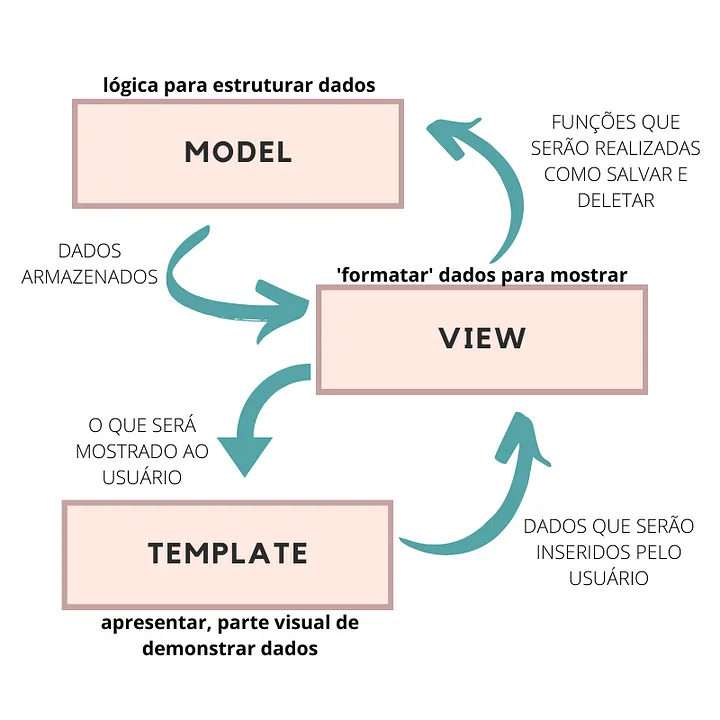
**Linha 9:** o objeto app chama uma método run( ) , dentro desse método ele passa como parâmetro um valor boolano para definir que o flask irá realizar atualizações em tempo real de execução, ou seja, o que você mudar no código não precisa derrubar o servidor e subir novamente, basta modificar o código e visualizar as modificações em seguida. Concluímos assim, nossa ambientação Flask.

# Ambientação templates

O flask entende o padrão MTV, onde dividimos aplicação em três partes. A primeira parte é M, que significa model. Nessa camada temos nossos códigos python orientado a objetos com funcionalidades (métodos) para cadastrar , atualizar, alterar e pesquisar no banco de dados. Já a letra “T” é a camada templates, nessa camada criamos nossas páginas web. A letra V, as views, são as controladoras da aplicação utilizada para tratar as requisições e respostas do nosso servidor web para nosso cliente (navegador – browser- páginas html).

Entender o funcionamento da arquitetura MTV do Flask/Django

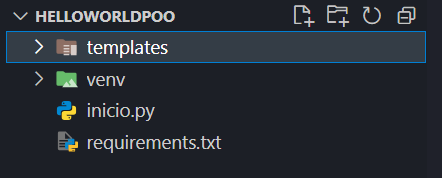
A imagem abaixo ilustra bem o funcionamento dessas camadas:



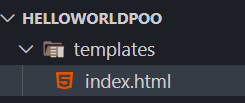
*Fonte:https://diandrasilva.medium.com/como-funciona-a-arquitetura-mtv-django-86af916f1f63*

Vamos iniciar pela ambientação da camada Template

Crie uma pasta chamada templates



Dentro dessa pasta, crie um arquivo index.html



Segue a codificação da página **index.html**

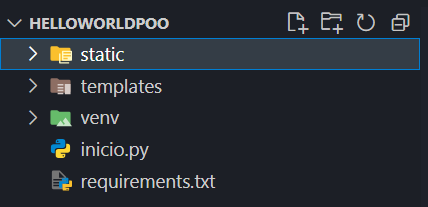


Você deve modificar a linha 2 para **“pt-br”** e também a linha 6 para **“Hello World POO”,** que respectivamente irá mudar o idioma e colocar um título na aplicação web.

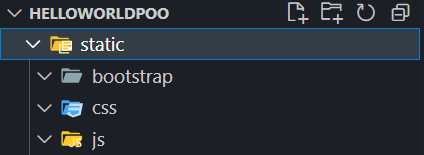
A aplicação web em seu **“front”** utiliza-se da linguagem html. Essa linguagem possui diversos componentes web. O html é considerado um “document”, ou seja, ele também é chamado de documento. No documento web definimos elementos. Esses elementos ou componentes web são os recursos para criar o aplicativo. Cada componente desse é criado a partir de um conjunto de tags.

<tag> </tag>

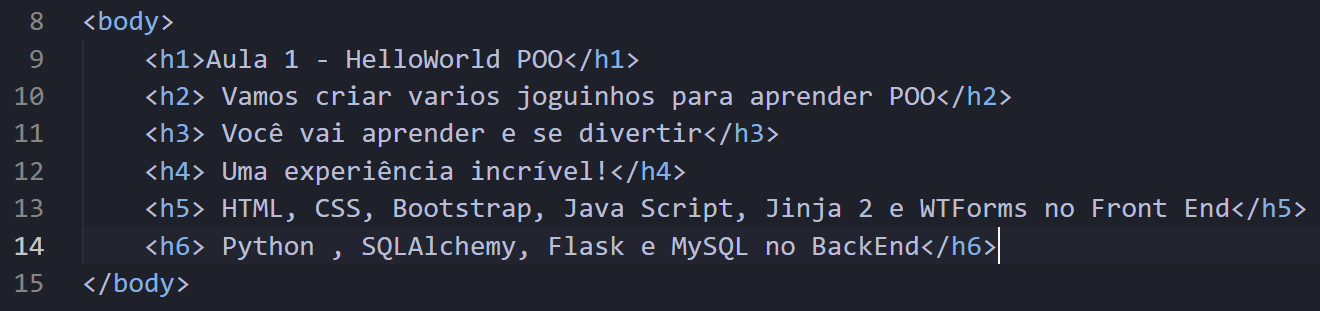
Para interagir com o JavaScript o Flask pede para criar uma pasta chamada static e nela colocar os seus códigos javascript, html, css e bootstrap.



Na pagina static vamos criar três pastas: css, javascript e bootstrap.



Vamos visualizar agora título em níveis html



**Na linha 8** – temos o corpo do nosso **aplicativo web**

**Na linha 15** – temos o corpo do nosso **aplicativo web**

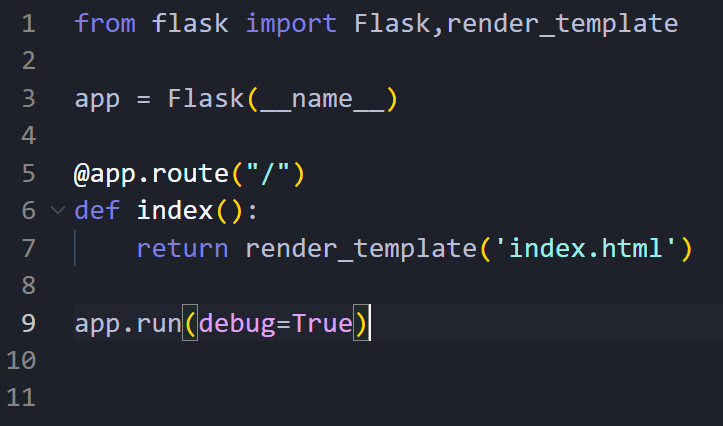
Da **linha 9 a 14** temos os níveis de cabeçalho.

Vamos agora renderizar nossa página.

Volte para o arquivo **inicio.py e importe render\_template , ele é responsável por renderizar páginas html na camada templates.**



Modifique sua rota “/” na função index( )



A **linha 7** teve uma modificação que é renderização da página index.html.

Agora execute a aplicação para visualizar a página renderizada.

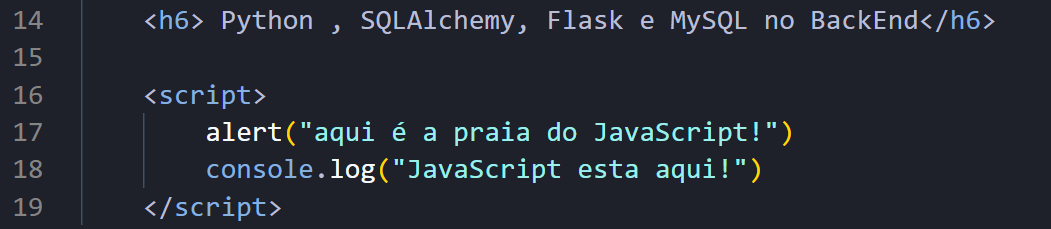


**E o JavaScript?**

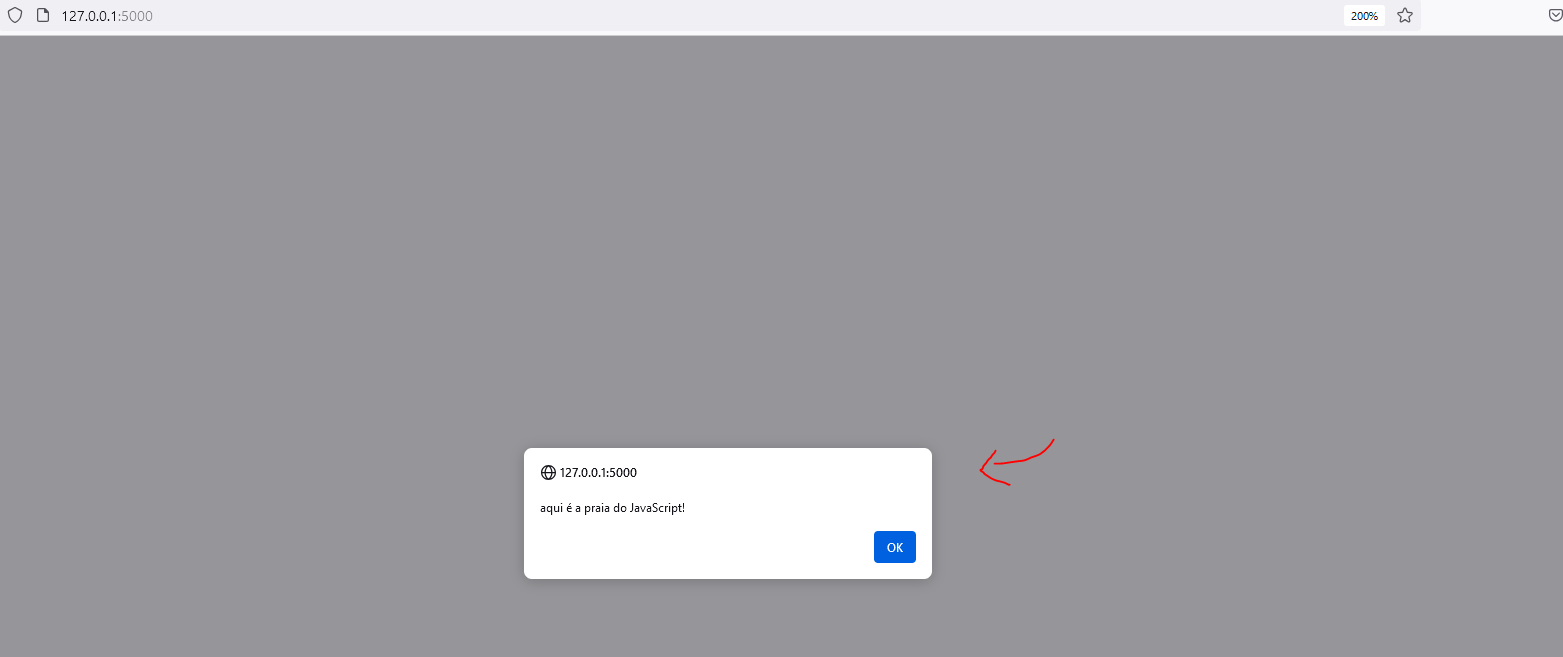
As interações dos jogos podem ser feitas com JavaScript. Podemos testar o uso do JavaScript usando a tag **<script>** dentro do nosso html.

Volte agora para nosso template, no arquivo **index.html.**

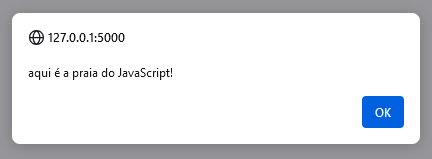
Abaixo do **<h6> </h6>** insira o seguinte código:



Execute a aplicação.



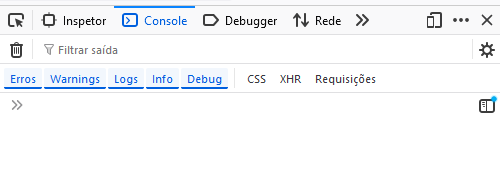
Abriu uma janela de alerta.



Vamos visualizar também inspecionando elemento nas ferramentas de desenvolvimento web.

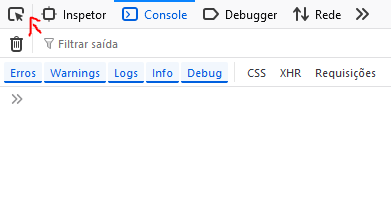
Utilizar Dev Tools dentro do navegador , inspecionando elementos html

Vamos abrir em um navegador, vou escolher o Firefox.

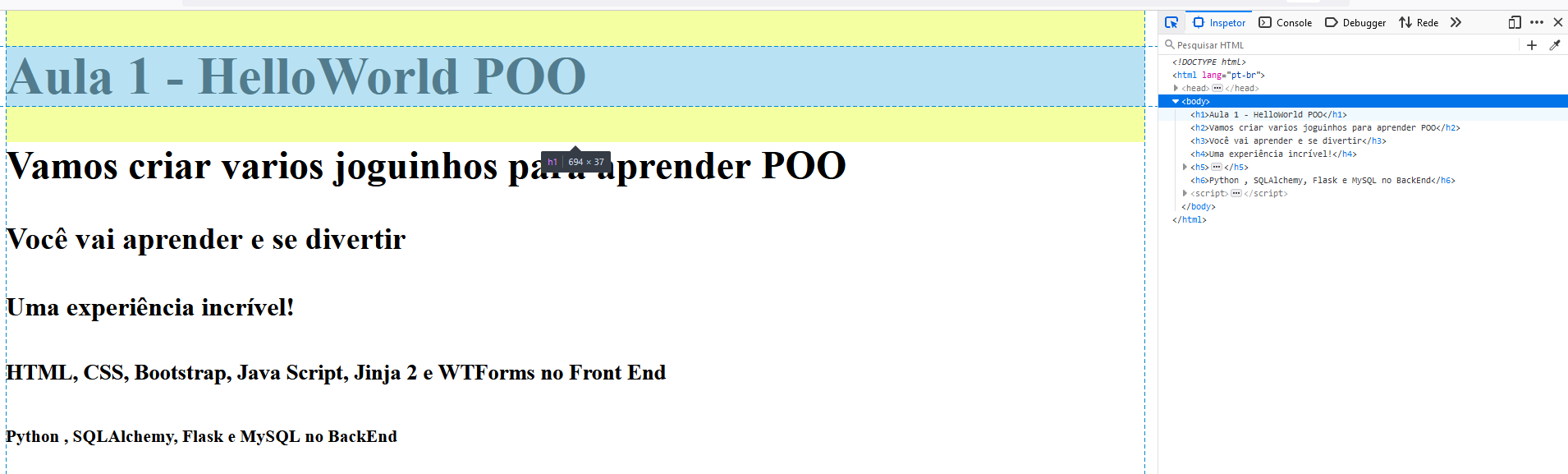


Para abrir o dev tools, pressione a tecla F12.

Clique na setinha que está marcada de vermelho



Ela irá inspecionar seu aplicativo web.



Vá passando o mouse em cima dos elementos, e ele vai mostrando todas as informações em tempo real do código. Leia as informações e viaje no mundo web.

Para visualizar a saída do comando console.log( ) você deve clicar na aba console.

Observe a aplicação do nosso código no inspecionador:

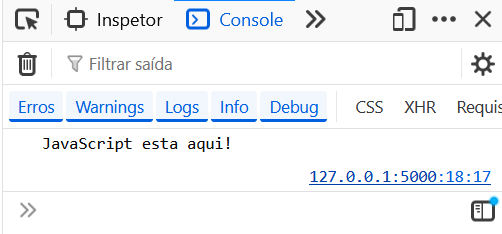
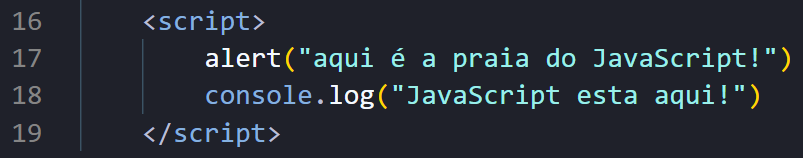
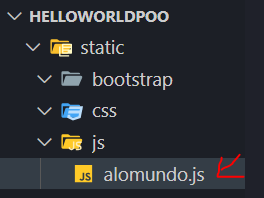
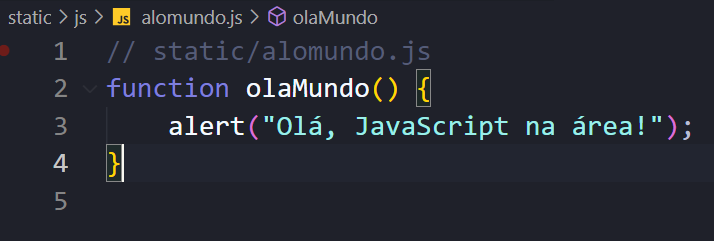
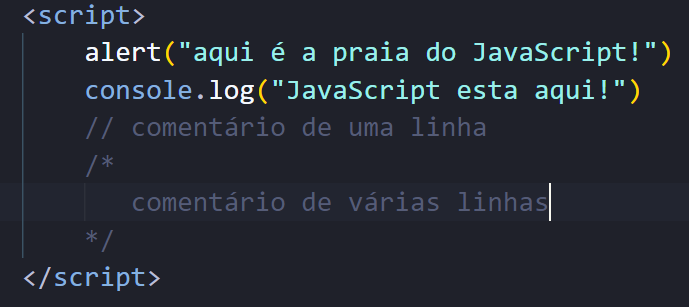


Figura - imagem do inspecionador e código a direita no Visual Studio Code , observe a linha 18.

Agora vamos pensar que essa mensagem esteja dentro de uma função javascript na pasta static.

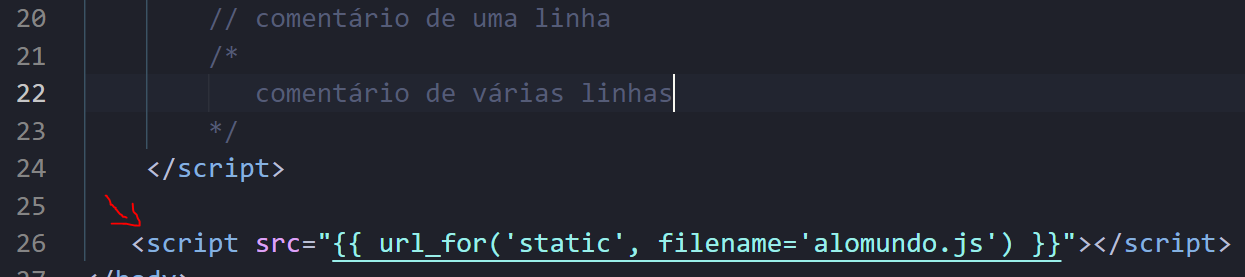


Lembra dos comentários?

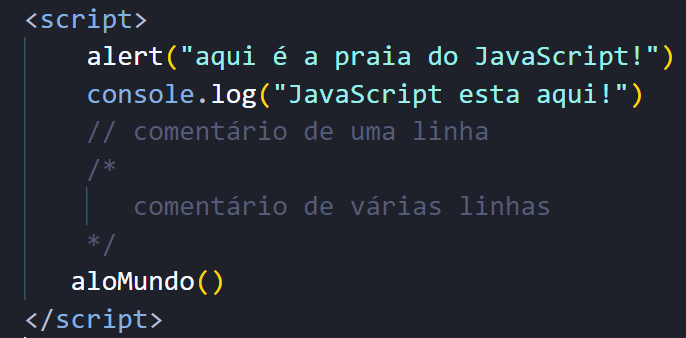


Veja como comentar no JavaScript.

No final desse <script> vamos adicionar um script que implementa o template engine do Flask o Jinja2. Nele podemos chamar a url que remete ao endereço da pasta static onde se encontra nossa função javascript.



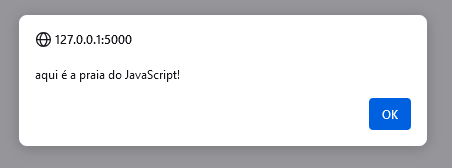
Agora podemos chamar nossa função JavaScript

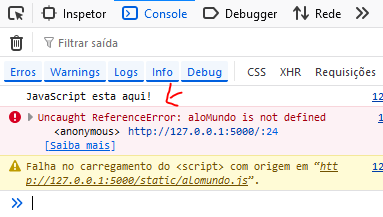


Vamos executar a aplicação:

1. Vai mostrar uma alert “aqui é a praia do JavaScript”
2. No Dev Tools do Firefox vai mostrar “Java Script esta aqui”
3. Por último, vai mostrar a mensagem “Olá, JavaScript esta na área”.

Vamos então executar:





Vimos um erro, dizendo que não esta reconhecendo a função aloMundo is not defined.

Essa área do dev tools vai mostrar tudo o que esta acontecendo seja mensagens de códigos, como mensagens de bugs.

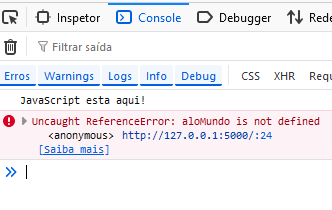
Observe que não foi passado todo o caminho para acessar o arquivo na pasta static



Vamos ajustar, e executar novamente:



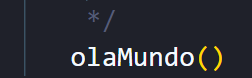
O erro persiste:



Enfrentar erros, é um dos desafios do programador de aplicativos web. Todo o dia ele enfrenta um novo erro, as vezes o mesmo erro.

É preciso atenção, na escrita, nas vírgulas, nas sintaxes, nas regras de nomenclatura. É preciso paciência e persistência para enfrentar e resolver os erros.

O erro foi encontrado, o nome da função tem a escrita errada:

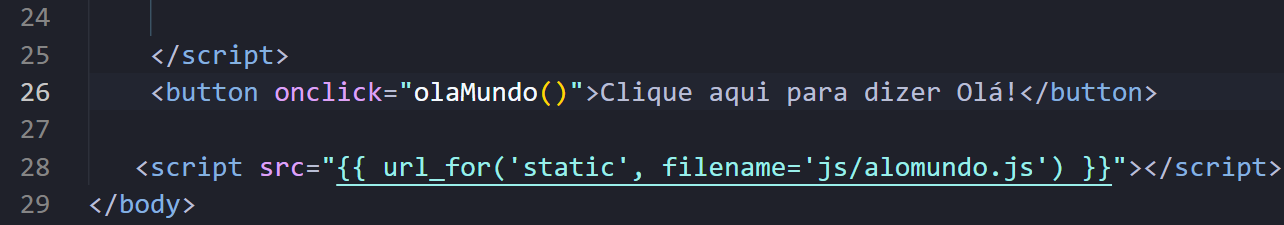


Vamos colocar essa chamada da função dentro de um botão. Isso também, pode acontecer no mundo real do trabalho. Você coloca um código em lugar e depois vê que é melhor mudar de lugar.

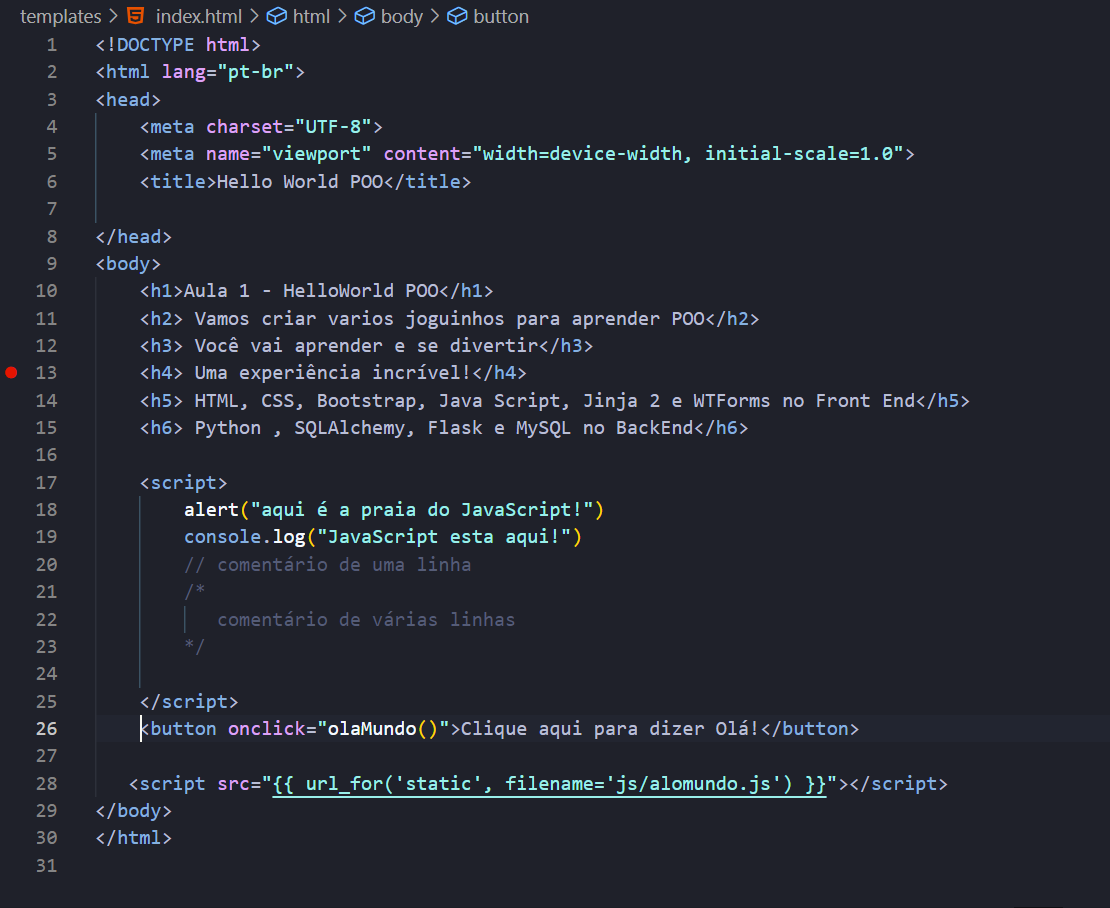


Para clicar no botão e mostrar o alert( ) crie o seguinte código:

Abaixo do script:



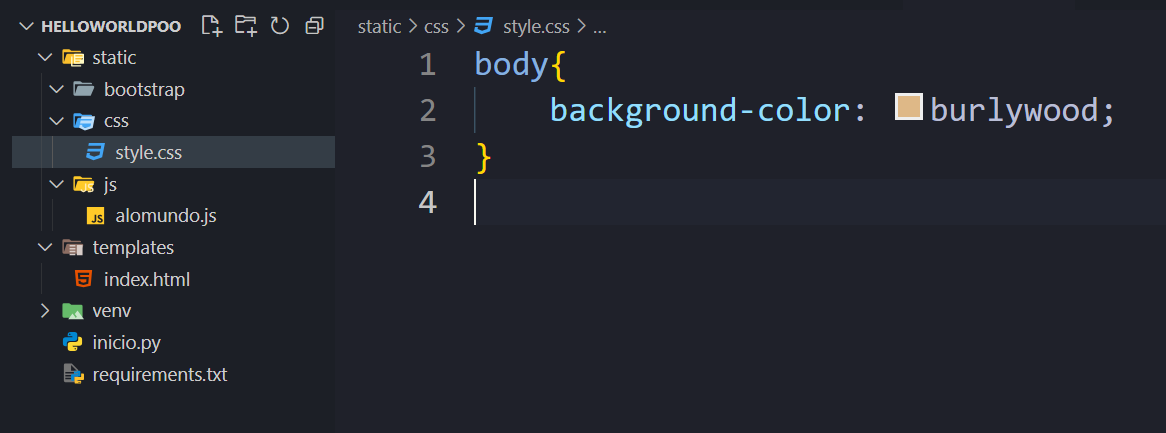
Para ter uma visão geral do código segue agora o código completo:



Diferenciar tecnologias web front-end com JavaScript e Jinja 2

**E o CSS?**

Precisamos também ambientar o nosso CSS no projeto. Para isso, crie um novo arquivo css.



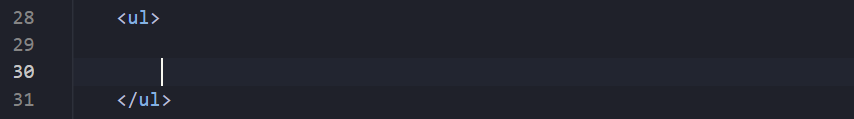
Voltemos para a página **index.html**

Paciência e resiliência para enfrentar bugs são softskills importantes para o profissional do futuro.

# DOM

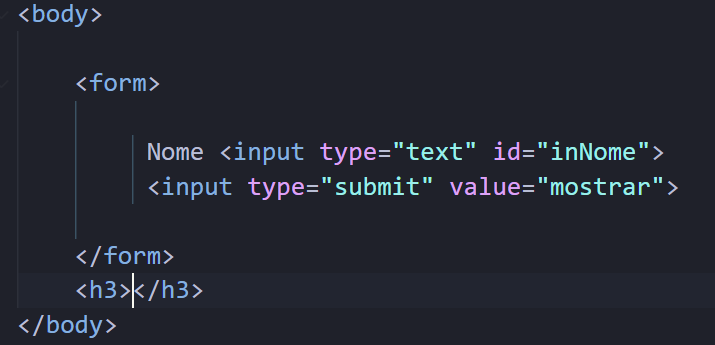
O **DOM** é uma interface de programação que os navegadores usam para interpretar e manipular documentos **HTML e XML**. Ele representa o documento como uma árvore de nós, onde cada nó é um objeto que representa uma parte do documento (como elementos, atributos e texto).

Abra o nosso arquivo crie uma lista:



Na lista insira um item que vai redirecionar para uma nova rota “/alovoce”

A rota **“/alovoce”** deverá ser redicionado para o arquivo **alovoce.html**

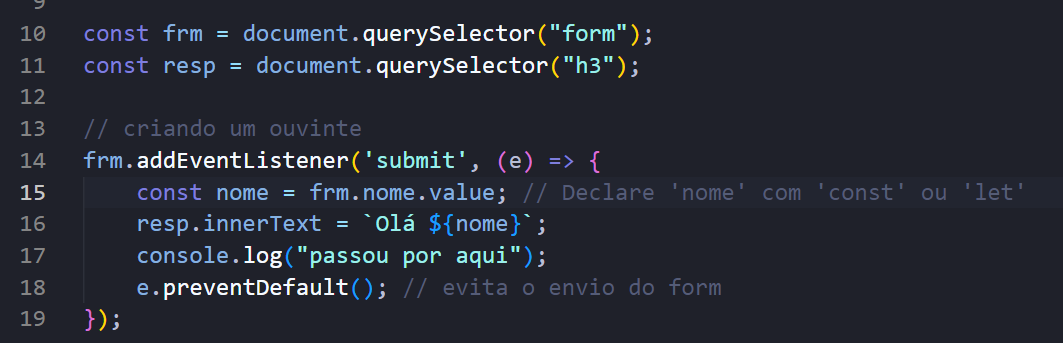


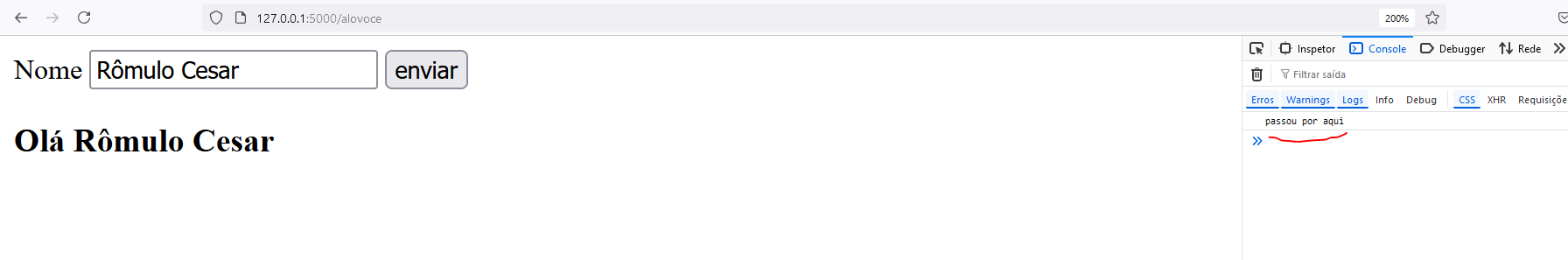
O <h3> </h3> será o elemento alvo do documento que será manipulado utilizando o conceito do **DOM**.

Vamos listar o código html:



Vamos listar o código javascript:







**E o CSS?**

O CSS é uma folha de estilo em cascata, ela serve para nos ajudar a estilizar páginas web. Vamos testar nossa estilização inicial?



Na página alô você, vamos inserir o código para incorporar o css à página.

Ao executar agora estará estilizado com as cores que você definiu.

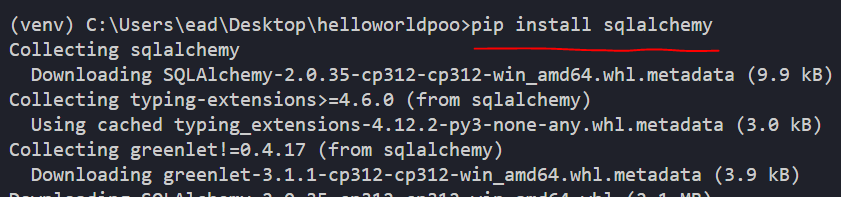


Vamos agora criar um cadastro de saladas. Para esse exemplo vamos criar uma nova página com o nome de cadsalada.html.

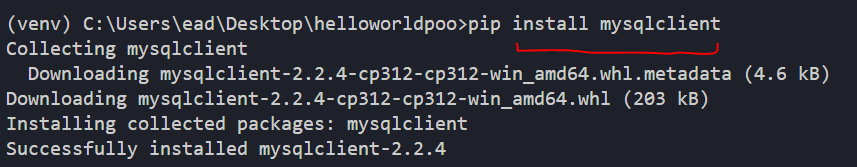
Para esse exemplo, utilizaremos as seguintes tecnologias

1. **SQLAlchemy** : banco de dados (ORM)
2. **MySQL**: SGBD
3. **Conectores:**  MySQL
4. **WTForms**: validador de formulário.

Vamos iniciar pela instalação do SQLAlchemy:



Também teremos que instalar o mysqlclient

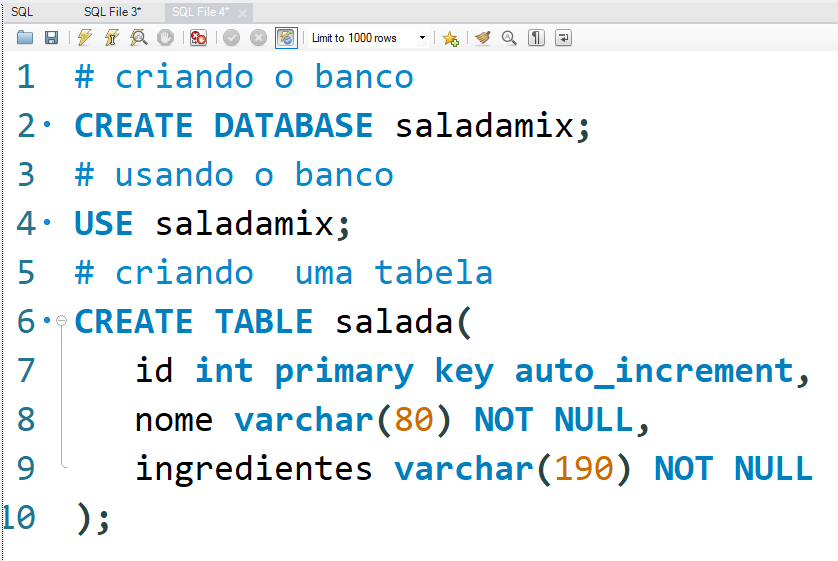


Vamos abrir o nosso banco de dados, o objetivo é criarmos um CRUD. O CRUD são as operações básicas de inserir, deletar, pesquisar e alterar no banco de dados.

Vamos usar também o SQLAlchemy para se comunicar com o banco de dados. Ele vai trabalhar com um mapeamento que irá transformar as tabelas em classes nesse mapeamento. Vai ficar bem legal!

Vamos abrir o MySQLWorkBench:

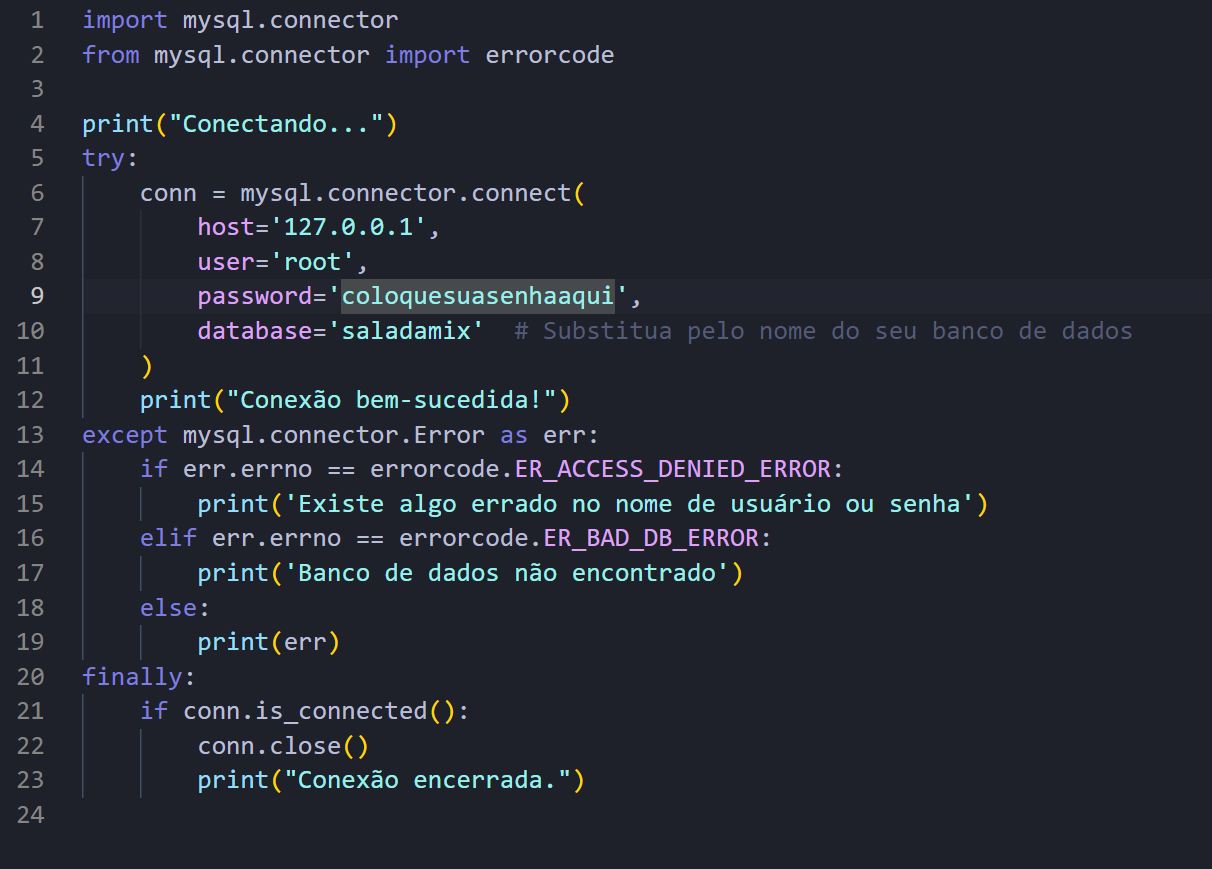
Crie o banco e a tabela a seguir:



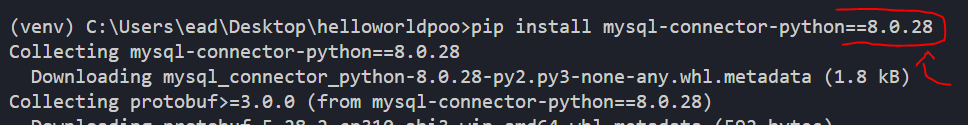
Agora crie um arquivo com o nome conexao.py



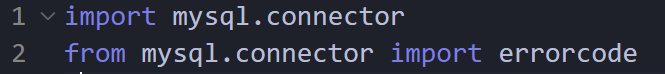
No arquivo **conexao.py** insira o seguinte código:



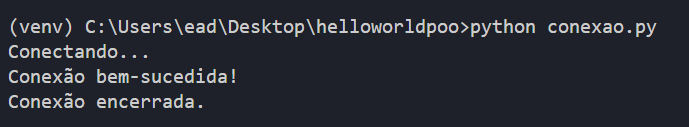
Instale o conector:



Agora importe essas duas linhas:



Vamos executar:



**import mysql.connector** # Importa o módulo **mysql.connector**, que fornece funções para **conectar-se** a um banco de dados MySQL.

from mysql.connector import errorcode # Importa a classe errorcode, que contém códigos de erro específicos do MySQL.

print("Conectando...") # Exibe uma mensagem no console indicando que o programa está tentando estabelecer uma conexão com o banco de dados.

try: # Inicia um bloco try para capturar possíveis exceções durante a tentativa de conexão ao banco de dados.

conn = mysql.connector.connect( # Chama a função connect() do módulo mysql.connector para estabelecer uma conexão.

host='127.0.0.1', # Define o endereço do host como '127.0.0.1', que é o endereço local (localhost).

user='root', # Define o nome de usuário para a conexão como 'root'.

password='coloquesuasenhaqui', # Define a senha para o usuário especificado.

database='saladamix' # Especifica o nome do banco de dados ao qual se deseja conectar.

)

print("Conexão bem-sucedida!") # Se a conexão for estabelecida com sucesso, exibe uma mensagem confirmando a conexão.

except mysql.connector.Error as err: # Captura exceções relacionadas ao MySQL e armazena o erro na variável 'err'.

if err.errno == errorcode.ER\_ACCESS\_DENIED\_ERROR: # Verifica se o erro é um problema de acesso (usuário ou senha incorretos).

print('Existe algo errado no nome de usuário ou senha') # Exibe uma mensagem de erro específica para acesso negado.

elif err.errno == errorcode.ER\_BAD\_DB\_ERROR: # Verifica se o erro é devido a um banco de dados que não existe.

print('Banco de dados não encontrado') # Exibe uma mensagem de erro indicando que o banco de dados especificado não foi encontrado.

else: # Se o erro não for um dos casos acima, trata-se de um erro diferente.

print(err) # Exibe a mensagem de erro genérica que foi capturada.

finally: # Inicia um bloco finally que sempre será executado, independentemente de um erro ter ocorrido ou não.

if conn.is\_connected(): # Verifica se a conexão foi estabelecida com sucesso (ou seja, se 'conn' está conectado).

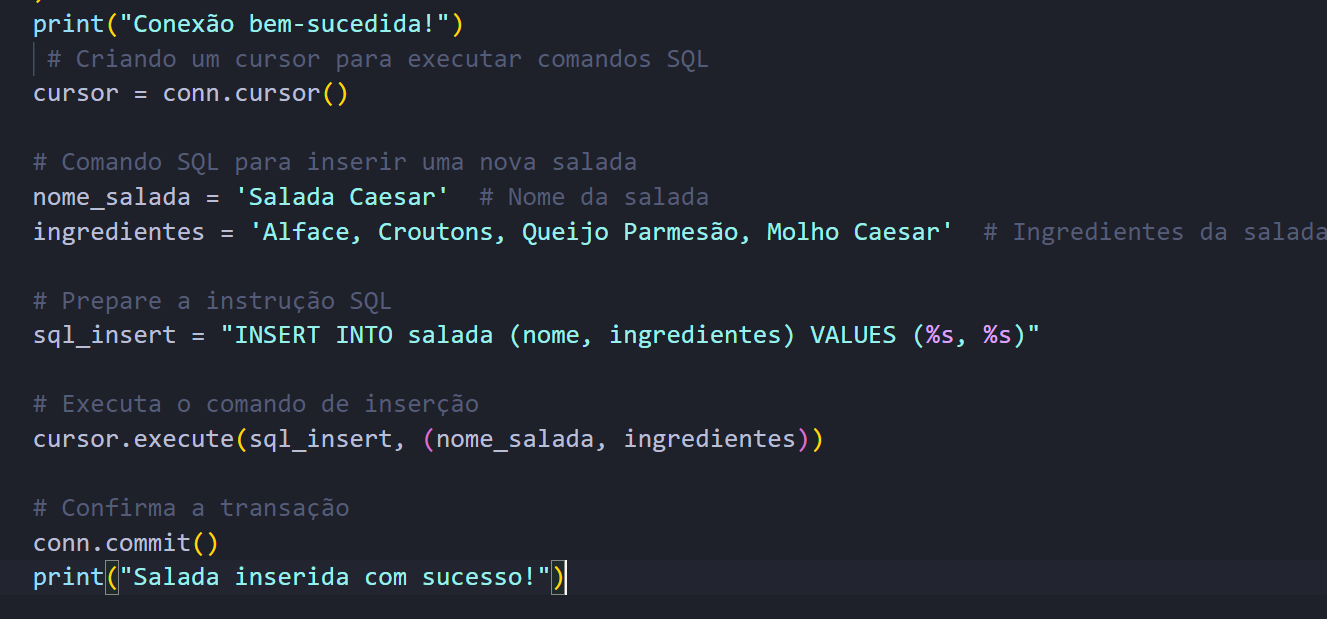
conn.close() # Se a conexão estiver ativa, fecha a conexão com o banco de dados para liberar recursos.

print("Conexão encerrada.") # Exibe uma mensagem informando que a conexão foi encerrada com sucesso.

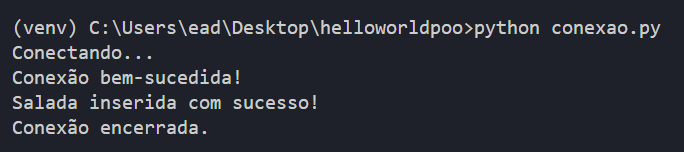
Entender e explicar sobre a conexão com um banco de dados relacional sem a utilização de frameworks de ORM, tratando exceções e personalizando mensagens para o usuário.

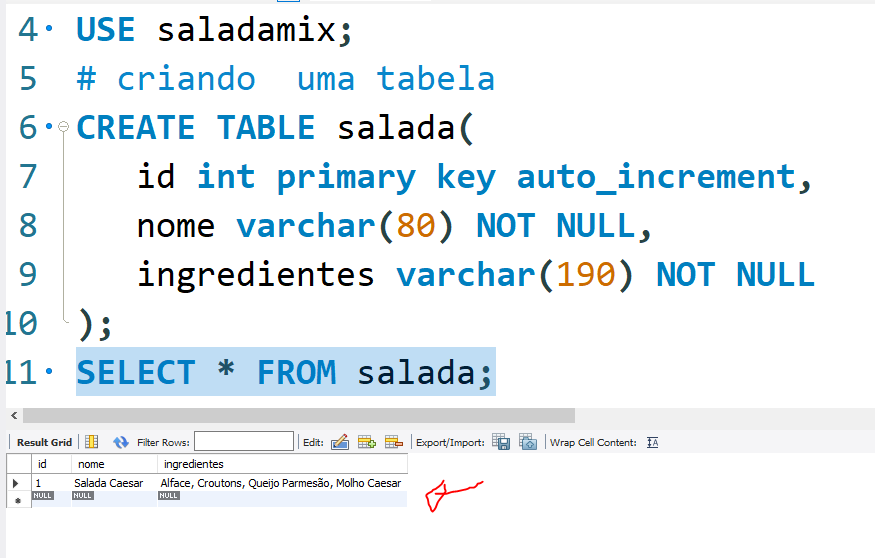
Vamos adicionar o seguinte código para testarmos a **inserção de valores**.

Vamos inserir:

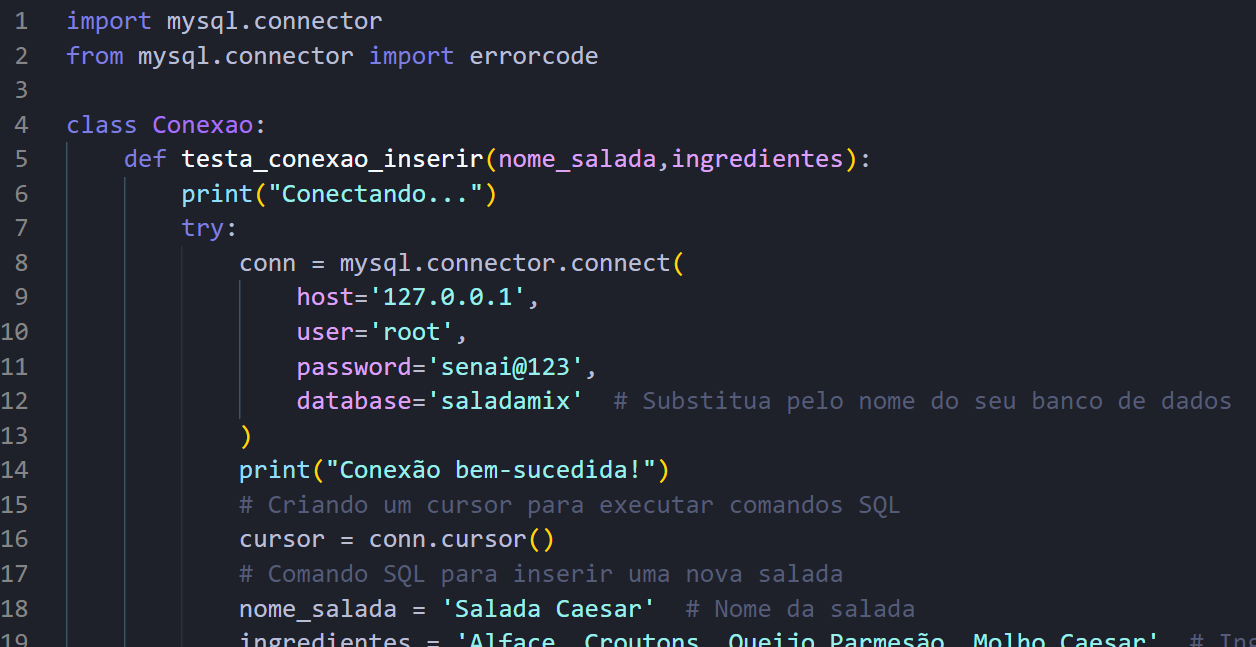


Vamos executar?





Muito bom! Vamos criar agora uma classe.



Agora vamos voltar para a nossas rotas, vamos criar uma rota e um template para tratar o cadastro de saladas.

Criar uma classe com **docstrings**